

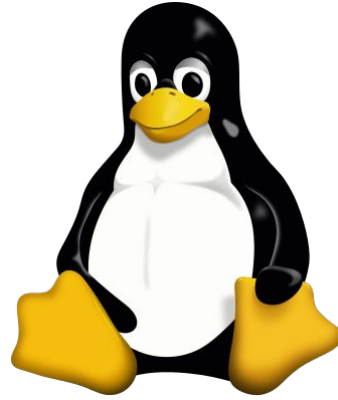
DIR-V GRAND CHALLENGE

VEGA Processor Tutorial

Building Linux for VEGA & Driver access

Centre for Development of Advanced Computing(C-DAC)

Nov 03, 2025

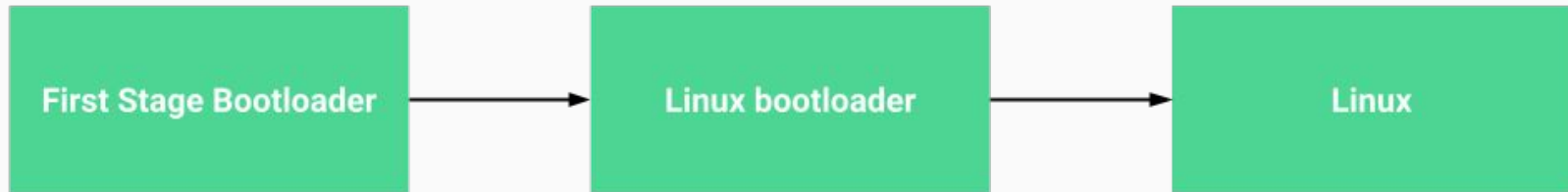


Building Linux for VEGA & Driver access

During the lectures

- ➡ Don't hesitate to ask questions. Other people in the audience may have similar questions too.
- ➡ This helps the trainer to detect any explanation that wasn't clear or detailed enough.
- ➡ Don't hesitate to share your experience, for example to compare Linux with other operating systems used in your company.
- ➡ Your point of view is most valuable, because it can be similar to your colleagues' and different from the trainer's.
- ➡ Your participation can make out session more interactive and make the topics easier to learn.

Linux boot flow



Bootloader

Available bootloaders :

- RISC-V Proxy Kernel and Boot Loader (riscv-pk)
- OpenSBI
- U-Boot

Here we are using riscv-pk

The RISC-V Proxy Kernel, pk, is a lightweight application execution environment that can host statically-linked RISC-V ELF binaries. It is designed to support tethered RISC-V implementations with limited I/O capability and thus handles I/O-related system calls by proxying them to a host computer.

bbl is expected to have been loaded from first stage boot loader, with the entry point running in machine mode. It is passed a device tree from the prior boot loader stage, and performs the following steps:

- ⇒ One hart is selected to be the main hart. The other harts are put to sleep until bbl is ready to transfer control to Linux, at which point they will all be woken up and enter Linux around the same time.
- ⇒ The device tree that was passed in from the previous stage is read and filtered.
- ⇒ All the other harts are woken up so they can set up their PMPs, trap handlers and enter supervisor mode.

- ➡ The mhartid CSR is read so Linux can be passed a unique per-hart identifier.
- ➡ Machine mode trap handlers, including a machine mode stack, is set up. bbl's machine mode code needs to handle both unimplemented instructions and machine-mode interrupts.
- ➡ The processor executes a mret to jump from machine mode to supervisor mode.
- ➡ bbl jumps to the start of its payload, which in this case is Linux.

Compiling Linux for TRISUL32

Buildroot

Buildroot is a set of Makefiles and patches that simplifies and automates the process of building a complete and bootable Linux environment for an embedded system, while using cross-compilation to allow building for multiple target platforms on a single Linux-based development system. Buildroot can automatically build the required cross-compilation toolchain, create a root file system, compile a Linux kernel image, and generate a boot loader for the targeted embedded system, or it can perform any independent combination of these steps. For example, an already installed cross-compilation toolchain can be used independently, while Buildroot only creates the root file system.

Buildroot for TRISUL32 : https://gitlab.com/dirv-vega-gc2025/trisul32_buildroot

For beginners, a simple build script **build.sh** is provided in **buildroot** directory. This will build Linux with predefined options.

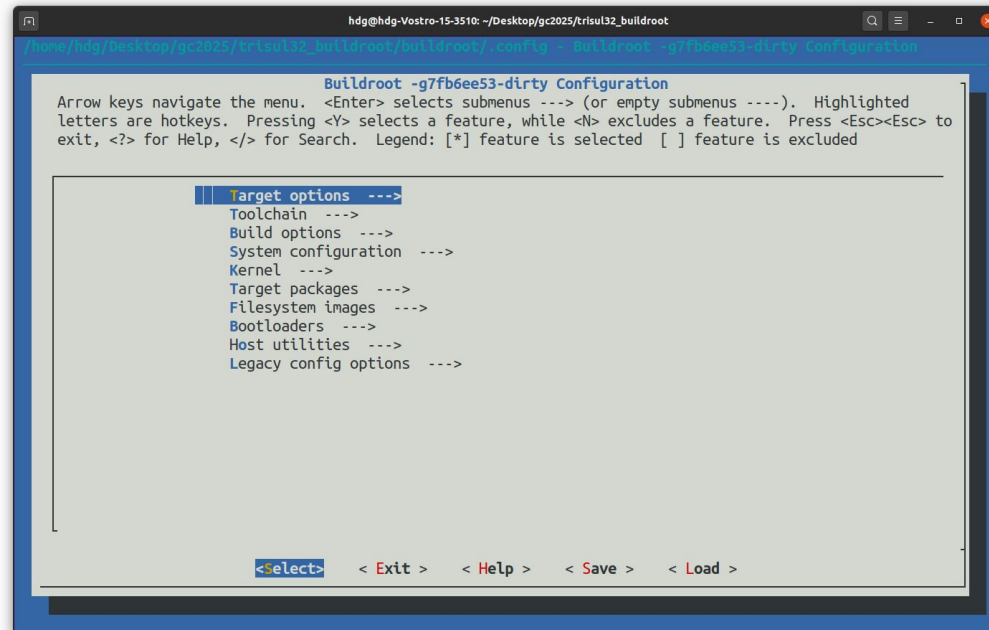
```
./build.sh
```

For people with little experience on Linux and buildroot, can try building Linux from scratch.

```
cd buildroot
```

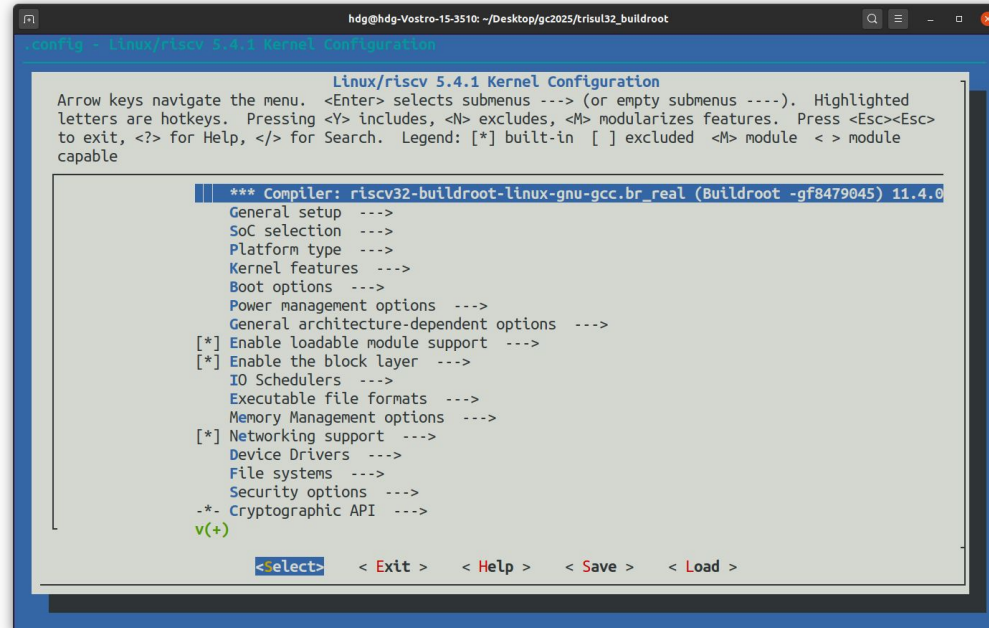
Buildroot menuconfig:

```
make menuconfig
```



Linux menuconfig:

```
make linux-menuconfig
```



The screenshot shows a terminal window titled "hdg@hdg-Vostro-15-3510: ~/Desktop/gc2025/trisul32_buildroot". The terminal displays the "Linux/riscv 5.4.1 Kernel Configuration" menu. At the top, instructions explain navigation: "Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [] excluded <M> module < > module capable". The menu items are listed as follows:

- *** Compiler: riscv32-buildroot-linux-gnu-gcc.br_real (Buildroot -gf8479045) 11.4.0
- General setup --->
- SoC selection --->
- Platform type --->
- Kernel features --->
- Boot options --->
- Power management options --->
- General architecture-dependent options --->
- [*] Enable loadable module support --->
- [*] Enable the block layer --->
- IO Schedulers --->
- Executable file formats --->
- Memory Management options --->
- [*] Networking support --->
- Device Drivers --->
- File systems --->
- Security options --->
- *- Cryptographic API --->

At the bottom, there is a green "v(+)" and a navigation bar with the following options: "<Select>", "< Exit >", "< Help >", "< Save >", and "< Load >".

After selecting necessary packages, lets build Linux

```
make -j8
```

If build is successful, output/images/ directory will contains final image, vmlinux.
Copy vmlinux to ../riscv-pk/build/. Now you could build Linux binary with Linux bootloader by :

```
../configure --host=riscv32-buildroot-linux-gnu --with-payload=./vmlinux  
--with-arch=rv32ima --with-abi=ilp32
```

```
make
```

```
riscv32-buildroot-linux-gnu-objcopy --gap-fill 0 -I elf32-littleriscv -O binary  
--set-section-flags .bss=alloc,load,contents bbl bbl.bin
```

```
hdg@hdg-Vostro-15-3510: ~/Desktop/gc2025/vega-tools/utills/e...  
-----  
Bootloader, ver 1.0.0 [ (hdg@cdac_tvm) Mon Jun 30 11:01:32 IST 2025 #160 ]  
-----  
ISA : RISC-V [RV32IMA]  
CPU : VEGA AT1051  
SoC : TRISUL32  
-----  
www.vegaprocessors.in | vega@cdac.in  
-----  
Press any key to choose the xmodem transfer  
  
The ethernet mode will be executed automatically in 00s.  
  
Transfer mode : Ethernet  
  
Memory : DDR2 [0x80000000 - 0x8fffffff] [256 MB]  
  
MAC: aa:bb:cc:dd:ee:ff  
  
Ethernet Initialised @ 100Mbps  
  
Waiting for data....
```

Edit the environment script in `vega-tools/utils/eth_transfer/` like below for transfer linux & DTS

```
#Number of files to send
files=2

#Addresses of corresponding files
address1=0x80000000
address2=0x20000

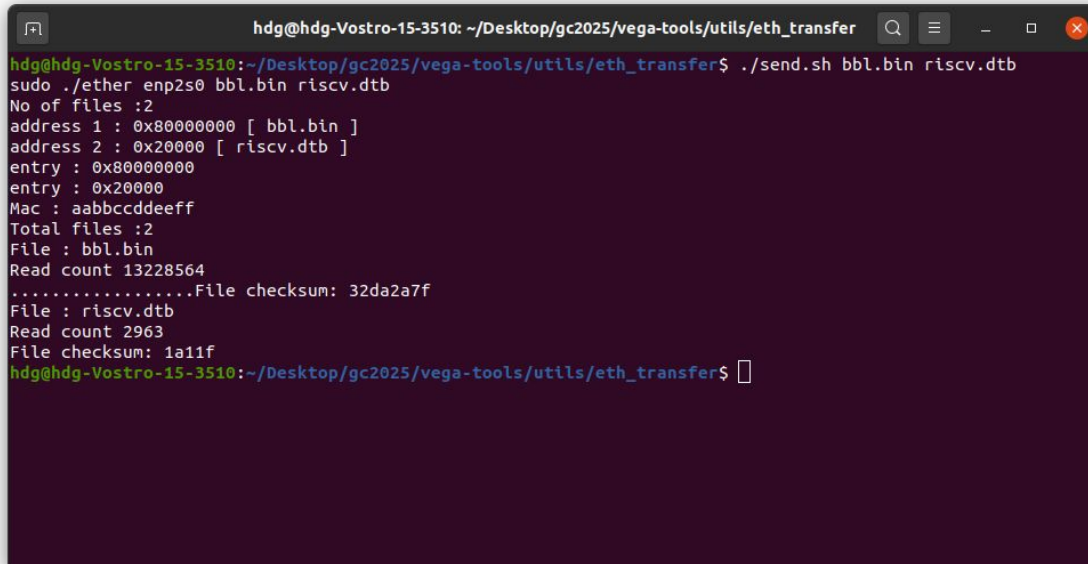
#Entrypoint of program to jump after download
entrypoint=0x80000000

#Device tree binary address
dtbaddress=0x20000

#MAC address of board to set
mac=aa:bb:cc:dd:ee:ff
```


send.sh script in vega-tools/utils/eth_transfer/ will transfer bbl.bin to board.

```
./send.sh bbl.bin riscv.dtb
```

A terminal window titled 'hdg@hdg-Vostro-15-3510: ~/Desktop/gc2025/vega-tools/utils/eth_transfer' showing the execution of the 'send.sh' script. The user enters './send.sh bbl.bin riscv.dtb'. The script outputs details about the files being transferred, including addresses, entry points, MAC addresses, and file checksums. The terminal text is as follows:

```
hdg@hdg-Vostro-15-3510:~/Desktop/gc2025/vega-tools/utils/eth_transfer$ ./send.sh bbl.bin riscv.dtb
sudo ./ether enp2s0 bbl.bin riscv.dtb
No of files :2
address 1 : 0x80000000 [ bbl.bin ]
address 2 : 0x20000 [ riscv.dtb ]
entry : 0x80000000
entry : 0x20000
Mac : aabbccddeeff
Total files :2
File : bbl.bin
Read count 13228564
.....File checksum: 32da2a7f
File : riscv.dtb
Read count 2963
File checksum: 1a11f
hdg@hdg-Vostro-15-3510:~/Desktop/gc2025/vega-tools/utils/eth_transfer$
```

After transfer completed, Linux will start booting.

```

hdg@hdg-Vostro-15-3510: ~/Desktop/gc2025/vega-tools/utlis/eth_transfer
[ 0.000000] CPU : VEGA AT1051
[ 0.000000] SoC : TRISUL32
[ 0.000000] www.vegaprocessors.in | vega@cdac.in
Press any key to choose the xmodem transfer

The ethernet mode will be executed automatically in 00s.
Transfer mode : Ethernet
Memory : DDR2 [0x80000000 - 0x8fffffff] [256 MB]
MAC: aa:bb:cc:dd:ee:ff
Ethernet Initialised @ 100Mbps

Waiting for data....
Download settings received

Total files : 2
File 1 address : 80000000
File 2 address : 20000
Entry point : 80000000

Waiting for file 1
.....
File received successfully

Total bytes : 13400596 ,[80000000] - [80cc7a14]

Waiting for file 2
.....
File received successfully

Total bytes : 2963 ,[20000] - [20b94]

bbl loader..
[ 0.000000] OF: fdt: Ignoring memory range 0x80000000 - 0x80400000
[ 0.000000] Linux version 5.4.1 (hdg@hdg-Vostro-15-3510) (gcc version 11.4.0 (Buildroot -gf8479045)) #12 SMP Mon Nov 3 10:52:54 IST 2025
[ 0.000000] initrd not found or empty - disabling initrd
[ 0.000000] Zone ranges:
[ 0.000000] Normal [mem 0x0000000000400000-0x0000000008fffffff]
[ 0.000000] Movable zone start for each node
[ 0.000000] Early memory node ranges
[ 0.000000] node 0: [mem 0x0000000000400000-0x0000000008fffffff]
[ 0.000000] Initmem setup node 0 [mem 0x0000000000400000-0x0000000008fffffff]
[ 0.000000] elf_hwcap is 0x1101
[ 0.000000] percpu: Embedded 12 pages/cpu s18380 r8192 d22580 u49152
[ 0.000000] Built 1 zonelists, mobility grouping on. Total pages: 63882
[ 0.000000] Kernel command line: console=hvc0 earlycon
[ 0.000000] Dentry cache hash table entries: 32768 (order: 5, 131072 bytes, linear)
  
```

Peripherals discussed today

- ⇒ Interrupt Controller (*PLIC*)
- ⇒ Ethernet (*10/100*)

PLIC

Linux Driver Location :

```
Device Drivers --->
```

```
↓
```

```
[*] CDAC Simple Interrupt Controller
```

Linux menuconfig snap shot :

```
IRQ chip support ----
```

```
[ ] RISC-V Interrupt Controller
```

```
[ ] SiFive Platform-Level Interrupt Controller
```

```
[ ] CDAC Platform-Level Interrupt Controller
```

```
[*] CDAC Simple Interrupt Controller
```

PLIC device tree node

```
plic0: interrupt-controller@ 20010000 {  
    #interrupt-cells = <1>;  
    compatible = "cdac,plic-1.0.0";  
    reg = <0x0 0x20010000 0x0 0x4000000 >;  
    riscv,ndev = <32>;  
    interrupt-controller;  
    interrupts-extended = <& CPU0_intc 9>;  
};
```

Linux Driver Log :

```
[ 0.000000] plic: mapped 32 interrupts to 1 (out of 1) handlers.
```

The /proc interface :

```
# cat /proc/interrupts
          CPU0
 1:         0   CDAC PLIC      6  10000600.spi
 3:         0   CDAC PLIC      8  10000800.i2c
 5:         0   CDAC PLIC      9  10000900.i2c
```

Ethernet

Linux Driver Location :

```
Device Drivers --->
↓
[*] Network device support --->
↓
[*] Ethernet driver support --->
↓
<*> CDAC ethernet ERMAC (10/100) on C-DAC devices
```

Linux menuconfig snap shot :

```
[ ] Broadcom devices
[ ] Cadence devices
[ ] Cortina Gemini devices
< > CDAC ethernet MAC on CDAC devices
<*> CDAC ethernet ERMAC (10/100) on C-DAC devices
< > Dave ethernet support (DNET)
[ ] EZchip devices
```

Ethernet device tree node

```
ethernet@20030000 {  
    device_type = "network";  
    compatible = "cdac,cdac_mac";  
    interrupt-parent = <&plic0>;  
    interrupts = <4>;  
    reg = <0x0 0x20030000 0x0 0x100>;  
};
```

Linux Driver Log :

```
[ 22.224000] CDAC ERMAC driver, version 1.00
[ 22.388000] libphy: cdac_mii_bus: probed
[ 22.388000] cdac_mac 20030000.ethernet eth0: using MII interface
[ 22.400000] Generic PHY 20030000.ethernet-ffffffff:01: attached PHY driver [Generic PHY]
(mii_bus:phy_addr=20030000.ethernet-ffffff)
[ 22.412000] cdac_mac 20030000.ethernet eth0: CDAC mac at 0x20030000 irq 1
```

List network interfaces using ip command :

```
# ip addr show
1: lo: <LOOPBACK> mtu 65536 qdisc noop qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_0
   link/ether aa:bb:cc:dd:ee:ff brd ff:ff:ff:ff:ff:ff
   inet 192.168.7.2/24 brd 192.168.7.255 scope global eth0
   valid_lft forever preferred_lft forever
```

Obtain IP using udhcpc command

```
# udhcpc
udhcpc: started, v1.31.1
[ 42.196000] Initializing ethernet....
udhcpc: sending discover
[ 43.976000] cdac_mac 20030000.ethernet eth0: link up (100/Half)
udhcpc: sending discover
udhcpc: sending select for 10.176.19.239
udhcpc: lease of 10.176.19.239 obtained, lease time 122
deleting routers
adding dns 10.176.0.11
```

To get information about network interface, eth0

```
# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:a0:22:01:01:8f brd ff:ff:ff:ff:ff:ff
    inet 10.176.19.239/23 brd 10.176.19.255 scope global eth0
        valid_lft forever preferred_lft forever
```

You could transfer program to board via scp.

```
scp hello root@10.176.19.239:/root/
```

Password for root user is root.

Hello Driver

A sample Hello driver code is provided in trisul32_buildroot folder, Edit the Makefile to compile the kernel module

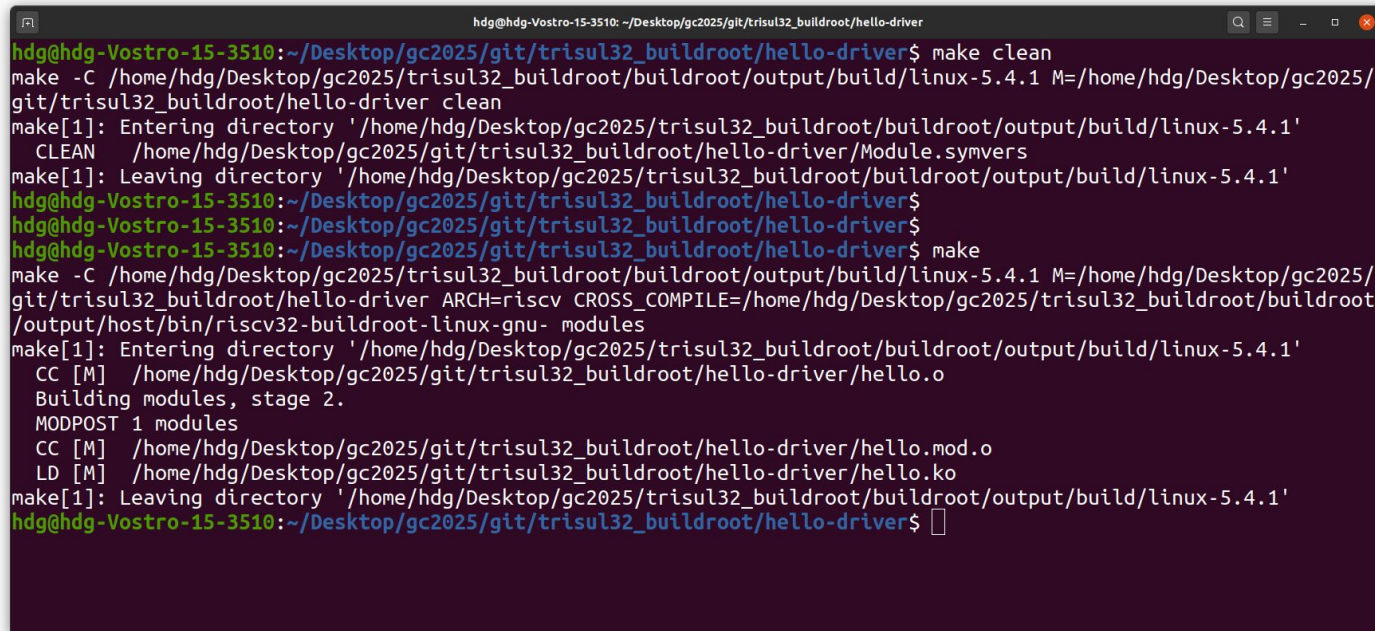
```
KDIR ?= /<path to linux source>/buildroot/output/build/linux-5.4.1
PWD  := $(shell pwd)

# Default target
obj-m += hello.o

all:
    $(MAKE) -C $(KDIR) M=$(PWD) ARCH=riscv CROSS_COMPILE=/<path to
    toolchain>/buildroot/output/host/bin/riscv32-buildroot-linux-gnu- modules

clean:
    $(MAKE) -C $(KDIR) M=$(PWD) clean
```

A sample Hello driver code is provided in trisul32_buildroot folder, Edit the Makefile to compile the kernel module

A terminal window with a dark background and light text. The window title is 'hdg@hdg-Vostro-15-3510: ~/Desktop/gc2025/git/trisul32_buildroot/hello-driver'. The user enters 'make clean' and 'make'. The output shows the cleaning of the build directory and the compilation of the 'hello.o' module. The final output shows the module 'hello.mod.o' is created and the 'hello.ko' file is generated.

```
hdg@hdg-Vostro-15-3510: ~/Desktop/gc2025/git/trisul32_buildroot/hello-driver
hdg@hdg-Vostro-15-3510:~/Desktop/gc2025/git/trisul32_buildroot/hello-driver$ make clean
make -C /home/hdg/Desktop/gc2025/trisul32_buildroot/buildroot/output/build/linux-5.4.1 M=/home/hdg/Desktop/gc2025/
git/trisul32_buildroot/hello-driver clean
make[1]: Entering directory '/home/hdg/Desktop/gc2025/trisul32_buildroot/buildroot/output/build/linux-5.4.1'
  CLEAN   /home/hdg/Desktop/gc2025/git/trisul32_buildroot/hello-driver/Module.symvers
make[1]: Leaving directory '/home/hdg/Desktop/gc2025/trisul32_buildroot/buildroot/output/build/linux-5.4.1'
hdg@hdg-Vostro-15-3510:~/Desktop/gc2025/git/trisul32_buildroot/hello-driver$
hdg@hdg-Vostro-15-3510:~/Desktop/gc2025/git/trisul32_buildroot/hello-driver$
hdg@hdg-Vostro-15-3510:~/Desktop/gc2025/git/trisul32_buildroot/hello-driver$ make
make -C /home/hdg/Desktop/gc2025/trisul32_buildroot/buildroot/output/build/linux-5.4.1 M=/home/hdg/Desktop/gc2025/
git/trisul32_buildroot/hello-driver ARCH=riscv CROSS_COMPILE=/home/hdg/Desktop/gc2025/trisul32_buildroot/buildroot
/output/host/bin/riscv32-buildroot-linux-gnu- modules
make[1]: Entering directory '/home/hdg/Desktop/gc2025/trisul32_buildroot/buildroot/output/build/linux-5.4.1'
  CC [M]   /home/hdg/Desktop/gc2025/git/trisul32_buildroot/hello-driver/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]   /home/hdg/Desktop/gc2025/git/trisul32_buildroot/hello-driver/hello.mod.o
  LD [M]   /home/hdg/Desktop/gc2025/git/trisul32_buildroot/hello-driver/hello.ko
make[1]: Leaving directory '/home/hdg/Desktop/gc2025/trisul32_buildroot/buildroot/output/build/linux-5.4.1'
hdg@hdg-Vostro-15-3510:~/Desktop/gc2025/git/trisul32_buildroot/hello-driver$
```


Copy [hello.ko](#) file to board. For that set a local IP for both board and PC

```
hdg@hdg-Vostro-15-3510: ~/Desktop/gc2025/vega-tools/eth_transfer

# ifconfig eth0 192.168.7.2
[ 126.403900] Initializing ethernet....
[ 127.285100] cdac_mac 20030000.ethernet eth0: link up (100)
#
#
# ifconfig
eth0      Link encap:Ethernet  HWaddr AA:BB:CC:DD:EE:FF
          inet addr:192.168.7.2  Bcast:192.168.7.255  Mask:20
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2 errors:0 dropped:1 overruns:0 frame:0
          TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:547 (547.0 B)  TX bytes:0 (0.0 B)
          Interrupt:4 Base address:0x7000

#
```

Board

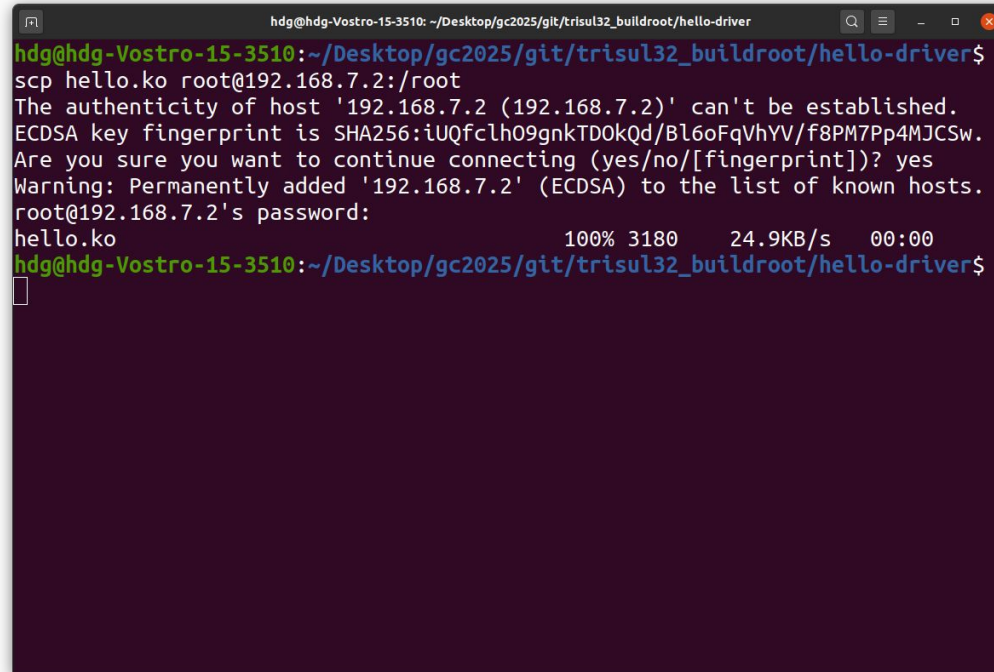
```
hdg@hdg-Vostro-15-3510:~$
hdg@hdg-Vostro-15-3510:~$
hdg@hdg-Vostro-15-3510:~$ sudo ifconfig enp2s0 192.168.7.1
[sudo] password for hdg:
hdg@hdg-Vostro-15-3510:~$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
        inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
        ether 02:42:be:55:d2:71 txqueuelen 0 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.7.1 netmask 255.255.255.0 broadcast 192.168.7.255
        ether b4:45:06:c6:66:b0 txqueuelen 1000 (Ethernet)
        RX packets 19134 bytes 1155358 (1.1 MB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 38672 bytes 54322261 (54.3 MB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
```

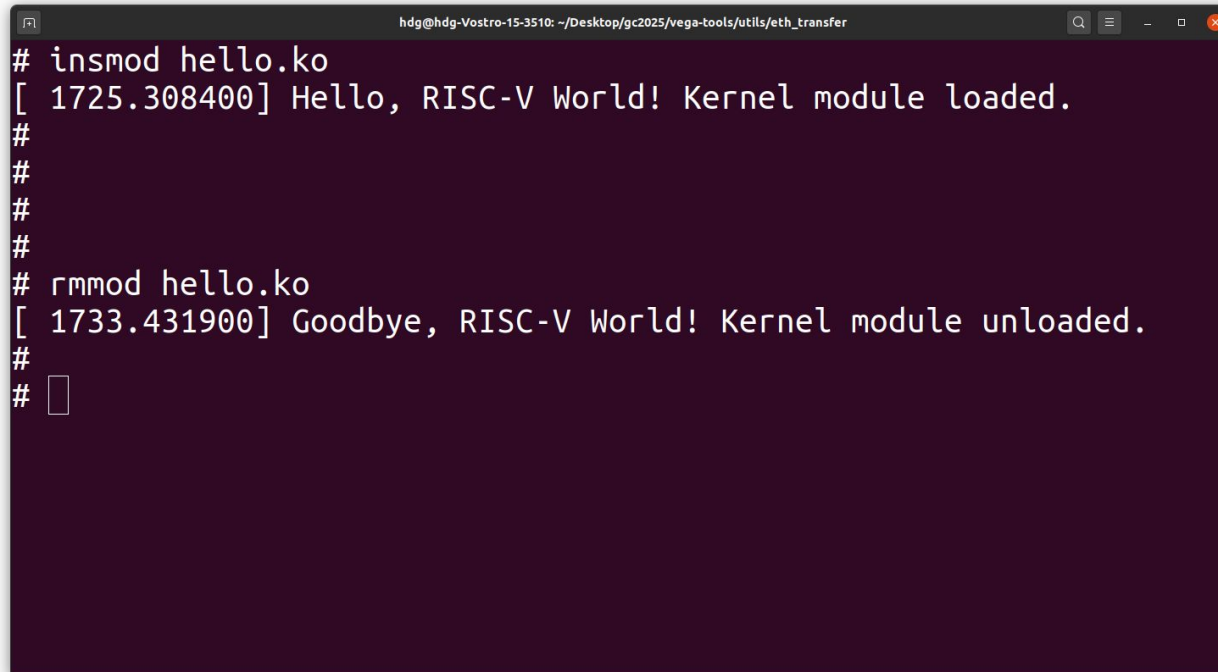
PC

Now send [hello.ko](#) using the scp command from PC.



```
hdg@hdg-Vostro-15-3510: ~/Desktop/gc2025/git/trisul32_buildroot/hello-driver
hdg@hdg-Vostro-15-3510:~/Desktop/gc2025/git/trisul32_buildroot/hello-driver$
scp hello.ko root@192.168.7.2:/root
The authenticity of host '192.168.7.2 (192.168.7.2)' can't be established.
ECDSA key fingerprint is SHA256:iUQfclh09gnkTDokQd/Bl6oFqVhYV/f8PM7Pp4MJCSw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.7.2' (ECDSA) to the list of known hosts.
root@192.168.7.2's password:
hello.ko                                100% 3180    24.9KB/s   00:00
hdg@hdg-Vostro-15-3510:~/Desktop/gc2025/git/trisul32_buildroot/hello-driver$
```

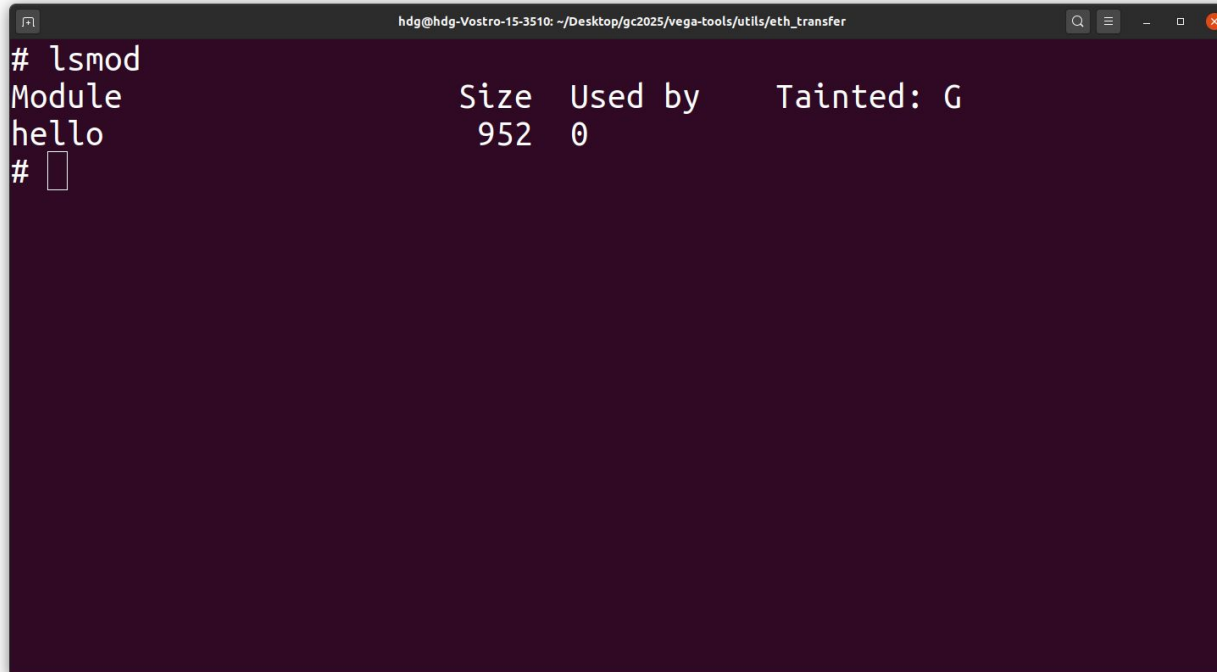
Insert the kernel module using **insmod** command and **rmmod** for remove the module

A terminal window with a dark purple background and white text. The window title bar shows the user 'hdg' on a 'hdg-Vostro-15-3510' machine, with the current directory being '~/.Desktop/gc2025/vega-tools/utlis/eth_transfer'. The terminal contains the following text:

```
# insmod hello.ko
[ 1725.308400] Hello, RISC-V World! Kernel module loaded.
#
#
#
#
#
# rmmod hello.ko
[ 1733.431900] Goodbye, RISC-V World! Kernel module unloaded.
#
#
```

A small white cursor is visible on the line following the last prompt.

List the available modules in Linux kernel by executing **lsmod** command

A terminal window with a dark purple background and white text. The window title bar shows the user 'hdg' on a 'Vostro-15-3510' machine, with the current directory being '~/Desktop/gc2025/vega-tools/utlis/eth_transfer'. The terminal content shows the command '# lsmod' has been executed, resulting in a table of loaded kernel modules. The table has four columns: 'Module', 'Size', 'Used by', and 'Tainted: G'. One module, 'hello', is listed with a size of 952 and is not used by any other process. The prompt '#' is followed by a cursor.

```
hdg@hdg-Vostro-15-3510: ~/Desktop/gc2025/vega-tools/utlis/eth_transfer
# lsmod
Module                Size  Used by    Tainted: G
hello                 952    0
#
```

Also check the **dmesg** for checking the kernel debug messages


```
hdg@hdg-Vostro-15-3510: ~/Desktop/gc2025/vega-tools/utils/eth_transfer
[ 47.978100] In cdac i2c added
[ 48.019300] NET: Registered protocol family 17
[ 48.049000] Key type ._fscrypt registered
[ 48.050400] Key type .fscrypt registered
[ 48.071900] cdac-pwm 10400000.pwm: C-DAC PWM Controller at 0x
[ 48.451200] Freeing unused kernel memory: 3032K
[ 48.452600] This architecture does not have kernel memory pro.
[ 48.459500] Run /init as init process
[ 126.403900] Initializing ethernet....
[ 127.285100] cdac_mac 20030000.ethernet eth0: link up (100/Ful)
[ 1557.099800] random: crng init done
[ 1701.122300] hello: loading out-of-tree module taints kernel.
[ 1701.159100] Hello, RISC-V World! Kernel module loaded.
[ 1716.895200] Goodbye, RISC-V World! Kernel module unloaded.
[ 1725.308400] Hello, RISC-V World! Kernel module loaded.
[ 1733.431900] Goodbye, RISC-V World! Kernel module unloaded.
#
```


Thanks!

Any questions?

You can find us at:

 youtube.com/vegaprocessors

 @vegaprocessor

 www.vegaprocessors.in

 vega@cdac.in